Attribution Models Compared

When marketing teams look at **attribution models compared** side by side, the discrepancies can be startling. A campaign that looks profitable under last-click often shows a loss under first-touch. The model you pick determines who gets the credit—and who gets the budget. That isn't a nuance. That's a financial lever.

The dirty secret? Most SaaS tools still default to last-click. It's computationally trivial. But last-click attribution for organic search is garbage if your sales cycle has any education or research phase. You need a strategy that matches how people actually buy. And for SEO specifically, you often need to rip apart server logs because JavaScript-based tracking misses bot visits, crawlers, and sometimes people too.

We'll kill the fluff. This document compares attribution models, but not with marketing slides. We'll implement them from raw access logs. You'll get real code, real data pipelines, and the kind of accuracy that only comes from ignoring the default dashboard.

What Attribution Models Actually Measure

An attribution model is a set of rules that splits conversion credit across touchpoints in a user's journey. That's it. Not machine learning fairy dust. Just division of a pie.

Every model answers the same question: which channel, keyword, or landing page should get the credit for a downstream action like a signup, purchase, or demo request? The disagreement between models isn't academic; it directly affects bidding decisions, content budget allocation, and whether your SEO team gets a pat on the back or a budget cut.

In SEO, we deal with a nasty twist. Organic search often acts as an assist—the first or second touch—and gets zero credit in a last-click world. A user searches, finds your blog, leaves, then later clicks a branded ad or direct visits and converts. Under last-click, search gets nothing. Under first-click, search gets everything. Neither captures reality. That's why implementing a custom model from server logs matters.

Why Server-Log Attribution Beats Cookie-Based Tracking for SEO Analysis

Cookie-pixel tracking is fragile. It fails under ITP, ad blockers, and cross-device jumps. Worse, third-party scripts often don't fire when Googlebot crawls your page, so the very entity that drives organic visibility leaves no trace in your analytics. If you rely on tag manager events, you're missing part of the story.

Server logs are ugly, huge, and brutally honest. They record every HTTP request, including Googlebot's user-agent string and the landing page path. You can reconstruct a clickstream without any JavaScript. That's crucial for SEO attribution because you can see the exact first organic entry point, not just the last conversion path from a pixel.

Before you run any model, strip out bot traffic. If you let Googlebot inflate your touch counts, you'll misattribute conversions to phantom visits. Here's how to grep out known crawlers from combined log format:

Extract real user hits for the conversion confirmation page, ignoring bots:

```
# grep -v removes lines matching known bot user agents
# We extract IP, timestamp, request path, status, referer, user-agent
grep -Eiv 'Googlebot|Bingbot|Slurp|YandexBot|DuckDuckBot' access.log \
  | grep ' 200 ' \
  | grep '/signup-thank-you' \
  | awk '{print $1, $4, $7, $11, $NF}' > clean_conversions.txt
```

When you skip bot filtering, your first-touch attribution might suddenly claim that "Googlebot" is your top-converting user agent. Not ideal.

Side-by-Side Credit Distribution (Table)

The differences become visceral when you put them in a table. This covers the six models you'll actually see in tools like Google Analytics, plus a custom log-based build.

Model	Credit Split	Best Use for SEO Content	Log Implementation Difficulty	Major Pitfall
First-click	100% to first touch	Top-of-funnel awareness content (blog posts, guides)	Medium – need to find earliest session start	Overvalues accidental discovery; ignores nurturing entirely
Last-click	100% to last touch before conversion	Branded search final step	Trivial – just the referrer before conversion page	Criminally undervalues SEO research touchpoints
Linear	Equal split to all touches	Long consideration cycles with multiple content pieces	Medium – count all sessions per user, divide evenly	Flattens influence; a blog view and a checkout have equal weight, which is wrong
Position-based (U-shaped)	40% first, 40% last, 20% split among middle	SEO as primary discovery plus last branded search	High – must classify touches as first/last/middle	Arbitrary 40-40-20 weights without data support
Time-decay	Closer to conversion gets more; earlier decays	When recent content refreshes drive conversions	Medium – compute half-life for each touch	Overestimates retargeting and brand ads if they cluster near conversion

Model	Credit Split	Best Use for SEO Content	Log Implementation Difficulty	Major Pitfall
Data-driven (Google Analytics 360)	Algorithmic based on Shapley values	Large datasets, but opaque and not available for free	Nearly impossible to replicate from logs without heavy stats	Black box; cannot debug why a touchpoint got 0.4% vs 5%
Log-based custom (our approach)	Rule-driven from raw server hits	When you need full control and bot-filtered truth	High – sessionization, dedup, path stitching	Complexity can bite you if session boundaries are wrong

I'll say it bluntly: for SEO reporting, first-click and position-based models give the least misleading view of organic's contribution. Last-click is corporate self-harm.

> Rule of thumb: Never attribute to a single click what multiple interactions can share — but only if you can reliably track the full path without bot noise.

Building a Log-Based Attribution Engine: Step-by-Step

We'll sessionize raw Apache/nginx logs, identify conversion events, and apply a linear model first (it's the clearest for demonstration). Then we'll tack on first-click and position-based as modifications. The core challenge is grouping sequential hits from the same IP+user-agent into visits, and then visits into a user journey over time using a temporary cookie or generated visitor ID.

Step 1: parse logs and filter out static file requests, image, CSS, JS, and bot user agents. You want only HTML page loads. Use a regex that captures IP, timestamp, method, path, referer, status, user-agent.

Python script to parse combined log format and extract relevant fields:

```
import re
from collections import defaultdict

LOG_PATTERN = re.compile(
    r'(\S+) \S+ \S+ \[(.*?)\] "\S+ (\S+) \S+" (\d+) \S+ "(.*?)" "(.*?)"'
)

def parse_line(line):
    match = LOG_PATTERN.search(line)
    if not match:
        return None
    ip, timestamp_str, path, status, referer, ua = match.groups()
    if status != '200':
        return None
    # ignore bots and static files
    if any(b in ua.lower() for b in ['bot','crawler','slurp']):
        return None
    if path.endswith(('.css','.js','.png','.jpg','.ico','.svg')):
        return None
    # crude session grouping: we'll use IP as visitor ID for demo—
    # in production, add cookie-based ID from %{COOKIEID}C
    return {'ip':ip, 'ts':timestamp_str, 'path':path, 'referer':referer, 'ua':ua}
```

Edge case: visitors behind CGNAT share IP. You'll need a cookie or a fingerprint to disambiguate, otherwise linear credit will smear across unrelated users.

Step 2: sort by IP and timestamp, then chunk into sessions using a 30-minute inactivity threshold. Mark the conversion page (e.g., /signup-confirmed) as the goal. For each user, find the list of session paths and referrers that led to the conversion.

```
# Assume parsed_log is a list of dicts sorted by ip, ts
from datetime import datetime, timedelta

SESS_TIMEOUT = timedelta(minutes=30)
GOAL_PATH = '/signup-confirmed'

def group_sessions(parsed_log):
    sessions = defaultdict(list)   # key: ip, value: list of sessions
    for entry in parsed_log:
```

```
        dt = datetime.strptime(entry['ts'], '%d/%b/%Y:%H:%M:%S %z')
        ip = entry['ip']
        # add to last session if within timeout
        if ip in sessions and sessions[ip]:
            last_sess = sessions[ip][-1]
            last_dt = last_sess[-1]['dt']
            if dt - last_dt < SESS_TIMEOUT:
                last_sess.append({**entry, 'dt':dt})
                continue
        # start new session
        sessions[ip].append([{**entry, 'dt':dt}])
    return sessions
```

This naive sessionization breaks when a user switches networks during a session. That's when the 30-min gap fails; you'll get two sessions for one true journey. Log-based attribution isn't perfect, but it's a hell of a lot more reliable than third-party cookie dropout.

Step 3: identify all users who reached the goal. Walk backwards through their sessions and collect touchpoints. A touchpoint is an entry path with its referrer. Distinguish organic search (referrer contains google, bing, etc.) for SEO flag.

```
def attribute_linear(converter_sessions):
    """Equal credit to all touchpoints across all sessions before conversion."""
    credit_map = {}
    for ip, sess_list in converter_sessions.items():
        all_touches = []
        # gather all sessions' landing pages and their referrers
        for sess in sess_list:
            for hit in sess:
                ref = hit['referer']
                if ref != '-':
                    # classify channel: organic, direct, referral, etc.
                    all_touches.append( (ref, hit['path']) )
        if not all_touches:
            continue
        share = 1.0 / len(all_touches)
        for ref, path in all_touches:
            key = (ref, path)
            credit_map[key] = credit_map.get(key, 0) + share
    return credit_map
```

Now you can sum credit per referring domain or per landing page. For search engines, you'd extract the domain from the referrer. The script above does a naive equal split; same logic applies for first-click (only first touch gets 1.0) or position-based.

Common Pitfalls When Implementing Attribution from Raw Logs

- **Bot traffic poisoning**: user-agent filtering must cover Googlebot, Bingbot, YandexBot, SemrushBot, AhrefsBot, and others. Miss one and bot "conversions" appear.
- **Missing cross-device stitching**: IP-based visitor IDs collapse mobile and desktop of the same person into separate users, deflating assist roles.
- **Inconsistent referrer field**: organic search referrers can be stripped by privacy settings. Then they appear as direct, cheapening SEO credit.
- **Session timeout too tight**: a 15-minute gap might split a single shopping session into two, inflating touchpoint count and diluting credit.
- **Ignoring non-200 status**: 301/302 redirects, 404 pages, or soft 404s appear as hits; you must exclude them, otherwise your conversion path includes garbage.

Real-World Example: Attributing a SaaS Signup to Organic Search Paths

In September 2024, an enterprise SaaS company we audited had a 6-month average sales cycle. Their analytics (last-click) showed that organic search drove only 12% of conversion credit. The VP of Marketing was about to slash the content team.

We pulled 18GB of Apache access logs, stripped bots, and ran a first-click linear model (ahem, custom log-based). The picture flipped. Organic search was the first touch for 43% of converted users. Those users discovered the product via long-tail guides, then later returned via direct or branded ads to finalize. When the credit distribution reflected actual

discovery, the SEO program got a massive budget increase – the right commercial decision.

Before: last-click attribution gave search 12% credit. After: first-touch log model gave search 43% credit. The raw data didn't change; the interpretation did.

FAQ: Attribution Models and Technical Constraints

Can I run attribution from Cloudflare logs instead of Apache?
Yes, but Cloudflare logs may lack the full referrer if a visitor comes from HTTPS. Set the Referrer-Policy header and use Cloudflare Workers to inject a custom cf-connecting-ip for session identification. You'll need to combine edge logs with origin logs for post-cache miss details.

How do I handle soft 404s that still return 200 OK?
Exclude pages with thin content by checking response body size in logs. If a page returns 200 but the bytes sent are less than 500 and the path isn't a redirect target, it's likely a soft 404. Filter those from conversion paths via a regex or size check in your parser.

Isn't data-driven better than any log implementation?
Data-driven requires Google Analytics 360 or similar machine learning platforms. It's opaque, expensive, and cannot be audited with logs. For teams that want to debug why search got 12% credit, a transparent log model wins. You can also use Shapley value approximation on log data if you have a team of data scientists, but that's overkill for most.

How do I verify that my bot filter list is complete?
Run a count of unique user agents from your log sample, and manually inspect the top 50. You'll often find obscure crawlers. Maintain a blacklist file. Also, use Google's crawler documentation to confirm official Googlebot IP ranges, then cross-reference against requests claiming to be Googlebot but originating from non-Google ASNs; many impersonators skew attribution.

Can I use this approach if my site is fully JavaScript-based (SPA)?
Server logs won't see SPA route changes unless you log each XHR or use a custom event beacon. You'll need to augment logs with front-end event tracking that pings a dedicated

endpoint. That endpoint will then appear in server logs as a hit, and you can extract paths from query parameters. Combine with log ingestion.

Pick a Model That Reflects Reality, Not Convenience

Attribution isn't about finding a "correct" model. It's about choosing a lens that aligns with your business cycle and then implementing it without data loss. Last-click is the easiest lens and the most distorted. First-click overcorrects. Linear muddies. Position-based might be the least bad default for SEO-heavy funnels.

But forget default. Implement your own from logs. You'll have to grapple with messy data, bot noise, and sessionization errors. But the moment you see your search team getting proper credit for a 4-month nurture path that closed a six-figure deal, the gunk is worth it. Just get the bot filter right first. And never let marketing dashboards hide the truth from the ones holding the server logs.

To submit and track large URL batches, a SpeedyIndex indexing service handles processing and status reporting.

Sources

1. Google. "Analytics Help Center." support.google.com/analytics
2. Google. "Campaign URL Builder (UTM)." ga-dev-tools.google
3. Nielsen Norman Group. "Conversion Rate Optimization." nngroup.com

---

About This Document

SpeedyIndex provides bulk indexing and index-checking tooling for agencies and affiliates managing thousands of URLs.

© 2026 SpeedyIndex. Reference document for educational use.